

AKD_PN_TG400_v05 [FB1]

AKD_PN_TG400_v05 Properties							
General							
Name	AKD_PN_TG400_v05	Number	1	Type	FB	Language	SCL
Numbering	automatic						
Information							
Title	AKD_PN_TG400_TIA13_v05	Author	jcoleman	Comment	This function block supports Profinet communication with the AKD drive using Telegram 400. It is written in TIA V13. For more information, please see the AKD Profinet Manual and the supporting documentation for this function block.	Family	TG400
Version	5.0	User-defined ID					

Name	Data type	Offset	Default value	Accessible from HMI	Visible in HMI	Setpoint	Comment
▼ Input							
iID	HW_IO	0.0	16#0	True	True	False	Hardware identifier
iInit	Bool	2.0	false	True	True	False	initialize config for TG400
iEStop	Bool	2.1	false	True	True	False	immediate quick stop and disable
iSW_Enable	Bool	2.2	false	True	True	False	enable/disable
iStop	Bool	2.3	false	True	True	False	stop all movement
iReset	Bool	2.4	false	True	True	False	reset drive faults
iStart_Homing	Bool	2.5	false	True	True	False	start homing
iStart_Jog	Bool	2.6	false	True	True	False	start jog
iStart_Move	Bool	2.7	false	True	True	False	start motion task
iMT_Pause	Bool	3.0	false	True	True	False	pause active motion task
iMT_MoveType	Word	4.0	16#0	True	True	False	motion task control word 0 = relative; 1 = absolute
iMTNum	Word	6.0	16#0	True	True	False	motion task number 0x8000 = direct motion task
iCmdPos	DWord	8.0	16#0	True	True	False	commanded position for motion task or homing
iCmdVel	DWord	12.0	16#0	True	True	False	commanded velocity for jog or direct motion task
iCmdAcc	Word	16.0	16#0	True	True	False	commanded acceleration for jog or direct motion task
iCmdDec	Word	18.0	16#0	True	True	False	commanded deceleration for jog or direct motion task
▼ Output							
oInitDone	Bool	20.0	false	True	True	False	initialization status - 0 while configuring TG400, 1= done
oStatus	DWord	22.0	16#0	True	True	False	ZSW1
oPos	DWord	26.0	16#0	True	True	False	actual position
oVel	Word	30.0	16#0	True	True	False	actual velocity x * 12000 rpm / 2^15
oCurrent	Word	32.0	16#0	True	True	False	actual current x * DRV.IPEAK / 2^14
oEnabled	Bool	34.0	false	True	True	False	Drive power on
oError	Bool	34.1	false	True	True	False	fault in function block or communication
oMoving	Bool	34.2	false	True	True	False	axis enabled and moving
oHomed	Bool	34.3	false	True	True	False	axis homed
oInPos	Bool	34.4	false	True	True	False	axis within in-position window
oFault	Bool	34.5	false	True	True	False	drive fault
oWarning	Bool	34.6	false	True	True	False	drive warning
InOut							
▼ Static							
▼ TG_400	"UDT_AKD_TG400"	36.0		True	True	False	
▼ Send	Struct	0.0		False	False	False	
STW1	Int	0.0	0	False	False	False	Control word 1
SATZANW	Int	2.0	0	False	False	False	Motion task selection
STW2	Int	4.0	0	False	False	False	Control word 2
MDI_TARPOS	Int	6.0	0	False	False	False	MDI motion task target position
MDI_TARPOS_1	Int	8.0	0	False	False	False	MDI motion task target position
MDI_VELOCITY	Int	10.0	0	False	False	False	MDI motion task velocity
MDI_VELOCITY_1	Int	12.0	0	False	False	False	MDI motion task velocity
MDI_ACC	Int	14.0	0	False	False	False	MDI motion task acceleration
MDI_DEC	Int	16.0	0	False	False	False	MDI motion task deceleration
MDI_MOD	Int	18.0	0	False	False	False	MDI motion task control word
HOME_DIST	Int	20.0	0	False	False	False	Home distance
HOME_DIST_1	Int	22.0	0	False	False	False	Home distance
IDT_DATAITEM_I_12	Int	24.0	0	False	False	False	Process data word input 13
IDT_DATAITEM_I_13	Int	26.0	0	False	False	False	Process data word input 14
IDT_DATAITEM_I_14	Int	28.0	0	False	False	False	Process data word input 15
IDT_DATAITEM_I_15	Int	30.0	0	False	False	False	Process data word input 16
▼ Receive	Struct	32.0		False	False	False	

Totally Integrated Automation Portal		
0103	COUNT := 64,	
0104	OUT => #bySDODataArray[0]);	
0105	#bSDOReq := TRUE;	
0106	#bSDOWriteDone := FALSE;	
0107	#wSDOStatus := 3; //proceed to Step 3	
0108	END_IF;	
0109	// setup the array data and start WRREC	
0110	IF #wSDOStatus = 3 THEN //Step 3	
0111	#bySDODataArray[0] := 16#fe; //Request Reference	
0112	#bySDODataArray[1] := 16#02; //2 for write PNU	
0113	#bySDODataArray[2] := 16#00; //Axis# is 0 for AKD	
0114	#bySDODataArray[3] := 16#01; // Number of parameters to write at a time	
0115	// Set PNU 916 subindex 0	
0116	#bySDODataArray[4] := 16#10; //Attribute is 0x10	
0117	#bySDODataArray[5] := 16#01; //Number of elements =1	
0118	#bySDODataArray[6] := 16#03; //0x394 for PNU 916	
0119	#bySDODataArray[7] := 16#94; //PNU 916	
0120	#bySDODataArray[8] := 16#00; //subindex	
0121	#bySDODataArray[9] := 16#00; //subindex 0 for 1st word of telegram	
0122	// To PNU 2 (ZSW1)	
0123	#bySDODataArray[10] := 16#42; //Format =0x42 for Word	
0124	#bySDODataArray[11] := 16#01; //Number of values	
0125	#bySDODataArray[12] := 16#00; //data	
0126	#bySDODataArray[13] := 16#02; //data - Signal #2 (0x02) for ZSW1	
0127		
0128	#SDOWrite(REQ := #bSDOReq, //do the PNU write	
0129	ID := #iID,	
0130	INDEX := 47, //AKD supports "Record Data 47"	
0131	LEN := 14, //data length	
0132	BUSY => #bSDOWriteBusy,	
0133	DONE => #bSDOWriteDone,	
0134	ERROR => #bSDOWriteError,	
0135	RECORD := #bySDODataArray);	
0136	IF #bSDOWriteBusy THEN	
0137	#bSDOReq := FALSE;	
0138	ELSIF #bSDOWriteDone THEN	
0139	#wSDOStatus := 4; //proceed to Step 4	
0140	END_IF;	
0141	END_IF;	
0142		
0143	// Clear the Array and start the request	
0144	IF #wSDOStatus = 4 THEN //Step 4	
0145	FILL_BLK(IN := 0,	
0146	COUNT := 64,	
0147	OUT => #bySDODataArray[0]);	
0148	#bSDOReq := TRUE;	
0149	#bSDOWriteDone := FALSE;	
0150	#wSDOStatus := 5; //proceed to Step 5	
0151	END_IF;	
0152	// setup the array data and start WRREC	
0153	IF #wSDOStatus = 5 THEN //Step 5	
0154	#bySDODataArray[0] := 16#fd; //Request Reference	
0155	#bySDODataArray[1] := 16#02; //2 for write PNU	
0156	#bySDODataArray[2] := 16#00; //Axis# is 0 for AKD	
0157	#bySDODataArray[3] := 16#01; // Number of parameters to write at a time	
0158	// Set PNU 916 subindex 1	
0159	#bySDODataArray[4] := 16#10; //Attribute is 0x10	
0160	#bySDODataArray[5] := 16#01; //Number of elements =1	
0161	#bySDODataArray[6] := 16#03; //0x394 for PNU 916	
0162	#bySDODataArray[7] := 16#94; //PNU 916	
0163	#bySDODataArray[8] := 16#00; //subindex	
0164	#bySDODataArray[9] := 16#01; //subindex 1 for 2nd word of telegram	
0165	// To PNU 33 (AKTSATZ)	
0166	#bySDODataArray[10] := 16#42; //Format =0x42 for Word	
0167	#bySDODataArray[11] := 16#01; //Number of values	
0168	#bySDODataArray[12] := 16#00; //data	
0169	#bySDODataArray[13] := 16#21; //data - Signal #33 (0x21) for AKTSATZ	
0170		
0171	#SDOWrite(REQ := #bSDOReq, //do the PNU write	
0172	ID := #iID,	
0173	INDEX := 47, //AKD supports "Record Data 47"	
0174	LEN := 14, //data length	
0175	BUSY => #bSDOWriteBusy,	
0176	DONE => #bSDOWriteDone,	
0177	ERROR => #bSDOWriteError,	
0178	RECORD := #bySDODataArray);	
0179	IF #bSDOWriteBusy THEN	
0180	#bSDOReq := FALSE;	
0181	ELSIF #bSDOWriteDone THEN	
0182	#wSDOStatus := 6; //proceed to Step 6	
0183	END_IF;	
0184	END_IF;	
0185		
0186	// Clear the Array and start the request	
0187	IF #wSDOStatus = 6 THEN //Step 6	
0188	FILL_BLK(IN := 0,	
0189	COUNT := 64,	
0190	OUT => #bySDODataArray[0]);	

Totally Integrated Automation Portal		
0191	#bSDOReq := TRUE;	
0192	#bSDOWriteDone := FALSE;	
0193	#wSDOStatus := 7; //proceed to Step 7	
0194	END_IF;	
0195	// setup the array data and start WRREC	
0196	IF #wSDOStatus = 7 THEN //Step 7	
0197	#bySDODataArray[0] := 16#fc; //Request Reference	
0198	#bySDODataArray[1] := 16#02; //2 for write PNU	
0199	#bySDODataArray[2] := 16#00; //Axis# is 0 for AKD	
0200	#bySDODataArray[3] := 16#01; // Number of parameters to write at a time	
0201	// Set PNU 916 subindex 2	
0202	#bySDODataArray[4] := 16#10; //Attribute is 0x10	
0203	#bySDODataArray[5] := 16#01; //Number of elements =1	
0204	#bySDODataArray[6] := 16#03; //0x394 for PNU 916	
0205	#bySDODataArray[7] := 16#94; //PNU 916	
0206	#bySDODataArray[8] := 16#00; //subindex	
0207	#bySDODataArray[9] := 16#02; //subindex 2 for 3rd word of telegram	
0208	// To PNU 4 (ZSW1)	
0209	#bySDODataArray[10] := 16#42; //Format =0x42 for Word	
0210	#bySDODataArray[11] := 16#01; //Number of values	
0211	#bySDODataArray[12] := 16#00; //data	
0212	#bySDODataArray[13] := 16#04; //data - Signal #4 (0x04) for ZSW2	
0213		
0214	#SDOWrite(REQ := #bSDOReq, //do the PNU write	
0215	ID := #iID,	
0216	INDEX := 47, //AKD supports "Record Data 47"	
0217	LEN := 14, //data length	
0218	BUSY => #bSDOWriteBusy,	
0219	DONE => #bSDOWriteDone,	
0220	ERROR => #bSDOWriteError,	
0221	RECORD := #bySDODataArray);	
0222	IF #bSDOWriteBusy THEN	
0223	#bSDOReq := FALSE;	
0224	ELSIF #bSDOWriteDone THEN	
0225	#wSDOStatus := 8; //proceed to Step 8	
0226	END_IF;	
0227	END_IF;	
0228		
0229	// Clear the Array and start the request	
0230	IF #wSDOStatus = 8 THEN //Step 8	
0231	FILL_BLK(IN := 0,	
0232	COUNT := 64,	
0233	OUT => #bySDODataArray[0]);	
0234	#bSDOReq := TRUE;	
0235	#bSDOWriteDone := FALSE;	
0236	#wSDOStatus := 9; //proceed to Step 9	
0237	END_IF;	
0238	// setup the array data and start WRREC	
0239	IF #wSDOStatus = 9 THEN //Step 9	
0240	#bySDODataArray[0] := 16#fb; //Request Reference	
0241	#bySDODataArray[1] := 16#02; //2 for write PNU	
0242	#bySDODataArray[2] := 16#00; //Axis# is 0 for AKD	
0243	#bySDODataArray[3] := 16#01; // Number of parameters to write at a time	
0244	// Set PNU 916 subindex 3	
0245	#bySDODataArray[4] := 16#10; //Attribute is 0x10	
0246	#bySDODataArray[5] := 16#01; //Number of elements =1	
0247	#bySDODataArray[6] := 16#03; //0x394 for PNU 916	
0248	#bySDODataArray[7] := 16#94; //PNU 916	
0249	#bySDODataArray[8] := 16#00; //subindex	
0250	#bySDODataArray[9] := 16#03; //subindex 3 for 4th word of telegram	
0251	// To PNU 28 (XIST_A)	
0252	#bySDODataArray[10] := 16#42; //Format =0x42 for Word	
0253	#bySDODataArray[11] := 16#01; //Number of values	
0254	#bySDODataArray[12] := 16#00; //data	
0255	#bySDODataArray[13] := 16#1C; //data - Signal #28 (0x1C) for XIST_A	
0256		
0257	#SDOWrite(REQ := #bSDOReq, //do the PNU write	
0258	ID := #iID,	
0259	INDEX := 47, //AKD supports "Record Data 47"	
0260	LEN := 14, //data length	
0261	BUSY => #bSDOWriteBusy,	
0262	DONE => #bSDOWriteDone,	
0263	ERROR => #bSDOWriteError,	
0264	RECORD := #bySDODataArray);	
0265	IF #bSDOWriteBusy THEN	
0266	#bSDOReq := FALSE;	
0267	ELSIF #bSDOWriteDone THEN	
0268	#wSDOStatus := 10; //proceed to Step 10	
0269	END_IF;	
0270	END_IF;	
0271		
0272	// Clear the Array and start the request	
0273	IF #wSDOStatus = 10 THEN //Step 10	
0274	FILL_BLK(IN := 0,	
0275	COUNT := 64,	
0276	OUT => #bySDODataArray[0]);	
0277	#bSDOReq := TRUE;	
0278	#bSDOWriteDone := FALSE;	

Totally Integrated Automation Portal		
<pre>0279 #wSDOStatus := 11; //proceed to Step 11 0280 END_IF; 0281 // setup the array data and start WRREC 0282 IF #wSDOStatus = 11 THEN //Step 11 0283 #bySDODataArray[0] := 16#fa; //Request Reference 0284 #bySDODataArray[1] := 16#02; //2 for write PNU 0285 #bySDODataArray[2] := 16#00; //Axis# is 0 for AKD 0286 #bySDODataArray[3] := 16#01; // Number of parameters to write at a time 0287 // Set PNU 916 subindex 4 0288 #bySDODataArray[4] := 16#10; //Attribute is 0x10 0289 #bySDODataArray[5] := 16#01; //Number of elements =1 0290 #bySDODataArray[6] := 16#03; //0x394 for PNU 916 0291 #bySDODataArray[7] := 16#94; //PNU 916 0292 #bySDODataArray[8] := 16#00; //subindex 0293 #bySDODataArray[9] := 16#04; //subindex 4 for 5th word of telegram 0294 // To PNU 6 (NIST_A) 0295 #bySDODataArray[10] := 16#42; //Format =0x42 for Word 0296 #bySDODataArray[11] := 16#01; //Number of values 0297 #bySDODataArray[12] := 16#00; //data 0298 #bySDODataArray[13] := 16#06; //data - Signal #6 (0x06) for NIST_A (VL.FB) 0299 0300 #SDOWrite(REQ := #bSDOReq, //do the PNU write 0301 ID := #iID, 0302 INDEX := 47, //AKD supports "Record Data 47" 0303 LEN := 14, //data length 0304 BUSY => #bSDOWriteBusy, 0305 DONE => #bSDOWriteDone, 0306 ERROR => #bSDOWriteError, 0307 RECORD := #bySDODataArray); 0308 IF #bSDOWriteBusy THEN 0309 #bSDOReq := FALSE; 0310 ELSIF #bSDOWriteDone THEN 0311 #wSDOStatus := 12; //proceed to Step 12 0312 END_IF; 0313 END_IF; 0314 0315 // Clear the Array and start the request 0316 IF #wSDOStatus = 12 THEN //Step 12 0317 FILL_BLK(IN := 0, 0318 COUNT := 64, 0319 OUT => #bySDODataArray[0]); 0320 #bSDOReq := TRUE; 0321 #bSDOWriteDone := FALSE; 0322 #wSDOStatus := 13; //proceed to Step 13 0323 END_IF; 0324 // setup the array data and start WRREC 0325 IF #wSDOStatus = 13 THEN //Step 13 0326 #bySDODataArray[0] := 16#f9; //Request Reference 0327 #bySDODataArray[1] := 16#02; //2 for write PNU 0328 #bySDODataArray[2] := 16#00; //Axis# is 0 for AKD 0329 #bySDODataArray[3] := 16#01; // Number of parameters to write at a time 0330 // Set PNU 916 subindex 5 0331 #bySDODataArray[4] := 16#10; //Attribute is 0x10 0332 #bySDODataArray[5] := 16#01; //Number of elements =1 0333 #bySDODataArray[6] := 16#03; //0x394 for PNU 916 0334 #bySDODataArray[7] := 16#94; //PNU 916 0335 #bySDODataArray[8] := 16#00; //subindex 0336 #bySDODataArray[9] := 16#05; //subindex 5 for 6th word of telegram 0337 // To PNU 52 (ITIST_GLATT) 0338 #bySDODataArray[10] := 16#42; //Format =0x42 for Word 0339 #bySDODataArray[11] := 16#01; //Number of values 0340 #bySDODataArray[12] := 16#00; //data 0341 #bySDODataArray[13] := 16#34; //data - Signal #52 (0x34) for ITIST_GLATT (IL.FB) 0342 0343 #SDOWrite(REQ := #bSDOReq, //do the PNU write 0344 ID := #iID, 0345 INDEX := 47, //AKD supports "Record Data 47" 0346 LEN := 14, //data length 0347 BUSY => #bSDOWriteBusy, 0348 DONE => #bSDOWriteDone, 0349 ERROR => #bSDOWriteError, 0350 RECORD := #bySDODataArray); 0351 IF #bSDOWriteBusy THEN 0352 #bSDOReq := FALSE; 0353 ELSIF #bSDOWriteDone THEN 0354 #wSDOStatus := 14; //proceed to Step 14 0355 //oInitDone := 1; //telegram configuration complete 0356 END_IF; 0357 END_IF; 0358 0359 // Set OpMode 0360 // Clear the Array and start the request 0361 IF #wSDOStatus = 14 THEN //Step 14 0362 FILL_BLK(IN := 0, 0363 COUNT := 64, 0364 OUT => #bySDODataArray[0]); 0365 #bSDOReq := TRUE; 0366 #bSDOWriteDone := FALSE;</pre>		

Totally Integrated Automation Portal		
<pre>0367 #wSDOStatus := 15; //proceed to Step 15 0368 END_IF; 0369 // Set OpMode = 2 Position 0370 IF #wSDOStatus = 15 THEN //Step 15 0371 #bySDODataArray[0] := 16#fe; //Request Reference 0372 #bySDODataArray[1] := 16#02; //2 for write PNU 0373 #bySDODataArray[2] := 16#00; //Axis# is 0 for AKD 0374 #bySDODataArray[3] := 16#01; // Number of parameters to write at a time 0375 // Set PNU 930 subindex 0 (DRV.OPMODE) 0376 #bySDODataArray[4] := 16#10; //Attribute is 0x10 0377 #bySDODataArray[5] := 16#01; //Number of elements =1 0378 #bySDODataArray[6] := 16#03; //0x3A2 for PNU 930 - opmode 0379 #bySDODataArray[7] := 16#A2; //PNU 930 0380 #bySDODataArray[8] := 16#00; //subindex 0 0381 #bySDODataArray[9] := 16#00; //subindex 0 0382 // To 2 (position mode) 0383 #bySDODataArray[10] := 16#42; //Format =0x42 for Word 0384 #bySDODataArray[11] := 16#01; //Number of values 0385 #bySDODataArray[12] := 16#02; //data 0386 #bySDODataArray[13] := 16#02; //data - vlaue = 2 for position mode 0387 0388 #SDOWrite(REQ := #bSDOReq, //do the PNU write 0389 ID := #iID, 0390 INDEX := 47, //AKD supports "Record Data 47" 0391 LEN := 14, //data length 0392 BUSY => #bSDOWriteBusy, 0393 DONE => #bSDOWriteDone, 0394 ERROR => #bSDOWriteError, 0395 RECORD := #bySDODataArray); 0396 IF #bSDOWriteBusy THEN 0397 #bSDOReq := FALSE; 0398 ELSIF #bSDOWriteDone THEN 0399 #wSDOStatus := 20; // setup complete - skip to process 0400 #oInitDone := TRUE; //telegram configuration complete 0401 END_IF; 0402 END_IF; 0403 END_IF; 0404 0405 // Read ZSW and set output values 0406 IF NOT #iInit AND #wSDOStatus > 19 THEN 0407 #wPNReadStatus := DPRD_DAT(LADDR := #iID, 0408 RECORD => #TG_400.Receive); 0409 IF #wPNReadStatus = 0 THEN 0410 #bPNReadError := FALSE; 0411 0412 #oStatus.%W0 := #TG_400.Receive.ZSW1; 0413 IF #wSDOStatus > 20 THEN 0414 #oStatus.%W1 := #wSDOReadValue; 0415 ELSE 0416 #oStatus.%W1 := 0; 0417 END_IF; 0418 #oPos.%W0 := #TG_400.Receive.XIST_A_1; 0419 #oPos.%W1 := #TG_400.Receive.XIST_A; 0420 #oVel := #TG_400.Receive.NIST_A; 0421 #oCurrent := #TG_400.Receive.ITIST_GLATT; 0422 0423 #oEnabled := #TG_400.Receive.ZSW1.%X0 AND #TG_400.Receive.ZSW1.%X1 AND #TG_400.Receive.ZSW1.%X2 AND NOT #TG_400.Receive.ZSW1.%X6; 0424 #oFault := #TG_400.Receive.ZSW1.%X3; 0425 #oMoving := NOT #TG_400.Receive.ZSW1.%X13 OR #TG_400.Receive.ZSW1.%X14 OR #TG_400.Receive.ZSW1.%X12; 0426 #oHomed := #TG_400.Receive.ZSW1.%X11; 0427 IF #TG_400.Send.STW1.%X6 THEN // Reset the InPos Flag for a minimum of one write cycle 0428 #oInPos := FALSE; 0429 ELSE 0430 #oInPos := #TG_400.Receive.ZSW1.%X10; 0431 END_IF; 0432 #oWarning := #TG_400.Receive.ZSW1.%X7; 0433 ELSE 0434 #bPNReadError := TRUE; 0435 END_IF; 0436 0437 END_IF; 0438 0439 // Do the write always as the zero Telegramm needs to be send on a init aswell 0440 #wPNWriteStatus := DPWR_DAT(LADDR := #iID, 0441 RECORD := #TG_400.Send); 0442 0443 // set STW1 and values for the AKD 0444 IF NOT #iInit AND #wSDOStatus > 19 THEN 0445 IF #wPNWriteStatus = 0 THEN 0446 #bPNWriteError := FALSE; 0447 0448 #TG_400.Send.STW1.%X10 := TRUE; 0449 IF #TG_400.Receive.ZSW1.%X9 THEN // PLC has control 0450 0451 // Do a EStop 0452 IF #iEStop THEN 0453 #TG_400.Send.STW1.%X2 := FALSE;</pre>		

Totally Integrated Automation Portal		
0454	#bEstop := TRUE;	
0455	END_IF;	
0456		
0457	// Do a reset	
0458	IF #iReset AND NOT #bResetDone THEN	
0459	#wSDOStatus := 20;	
0460	#TG_400.Send.STW1.%X7 := TRUE;	
0461	#bResetDone := TRUE;	
0462	#bEstop := FALSE;	
0463	#wStartReset := #TG_400.Receive.ZSW2;	
0464	END_IF;	
0465	IF ABS(#wStartReset - #TG_400.Receive.ZSW2) > 2 THEN	
0466	#TG_400.Send.STW1.%X7 := FALSE;	
0467	END_IF;	
0468		
0469	// Enable the drive	
0470	IF #iSW_Enable AND NOT #bEstop AND NOT (#bSWEnabled AND #TG_400.Receive.ZSW1.%X6 OR #TG_400.Re-	
ceive.ZSW1.%X3) THEN		
0471	IF #TG_400.Receive.ZSW1.%X6 THEN // Switch on inhibited	
0472	#TG_400.Send.STW1.%X1 := TRUE;	
0473	#TG_400.Send.STW1.%X2 := TRUE;	
0474	ELSE	
0475	IF #TG_400.Receive.ZSW1.%X0 THEN // Ready for switching on	
0476	#TG_400.Send.STW1.%X0 := TRUE;	
0477	IF #TG_400.Receive.ZSW1.%X1 THEN // Switched on	
0478	#TG_400.Send.STW1.%X3 := TRUE;	
0479	IF #TG_400.Receive.ZSW1.%X2 THEN // Operation	
0480	#TG_400.Send.STW1.%X12 := TRUE; //turn on real time jogging mode	
0481	#bSWEnabled := TRUE;	
0482		
0483	// Do a Stop	
0484	IF #iStop THEN	
0485	#TG_400.Send.STW1.%X4 := FALSE; // Stop active motion task	
0486	#TG_400.Send.STW1.%X5 := FALSE; // Stop active motion task	
0487	#TG_400.Send.STW1.%X8 := FALSE; // Stop jog 1	
0488	#TG_400.Send.STW1.%X9 := FALSE; // Stop jog 2	
0489	#TG_400.Send.STW1.%X11 := FALSE; // Stop homeing	
0490	//#TG_400.Send.STW1.%X12 := FALSE; // Stop real time jogging mode	
0491	ELSE	
0492		
0493	// Do home	
0494	IF #iStart_Homing AND NOT #bHomeStarted THEN	
0495	#TG_400.Send.HOME_DIST := #iCmdPos.%W1;	
0496	#TG_400.Send.HOME_DIST_1 := #iCmdPos.%W0;	
0497	#TG_400.Send.STW1.%X11 := TRUE;	
0498	#wHomingStartPoint := #TG_400.Receive.ZSW2;	
0499	#bHomeStarted := TRUE;	
0500	ELSE	
0501	IF #TG_400.Receive.ZSW1.%X11 AND #TG_400.Receive.ZSW1.%X13 AND (ABS(#wHoming-	
StartPoint - #TG_400.Receive.ZSW2) > 5) THEN		
0502	#TG_400.Send.STW1.%X11 := FALSE;	
0503	END_IF;	
0504		
0505	// Jogging	
0506	IF #iStart_Jog AND #iCmdVel <> 0 THEN	
0507	#TG_400.Send.STW1.%X8 := TRUE;	
0508	//#TG_400.Send.STW1.%X12 := TRUE;	
0509		
0510	// if the speed input is negative set Bit 13 for reverse direction	
0511	IF #iCmdVel.%X31 = 1 THEN	
0512	#TG_400.Send.STW1.%X13 := TRUE;	
0513	#TG_400.Send.MDI_VELOCITY := (16#ffff - #iCmdVel.%W1 + 1);	
0514	#TG_400.Send.MDI_VELOCITY_1 := (16#ffff - #iCmdVel.%W0 + 1);	
0515	ELSE	
0516	#TG_400.Send.STW1.%X13 := FALSE;	
0517	#TG_400.Send.MDI_VELOCITY := #iCmdVel.%W1;	
0518	#TG_400.Send.MDI_VELOCITY_1 := #iCmdVel.%W0;	
0519	END_IF;	
0520		
0521	#TG_400.Send.MDI_ACC := #iCmdAcc;	
0522	#TG_400.Send.MDI_DEC := #iCmdDec;	
0523	ELSE	
0524	#TG_400.Send.STW1.%X8 := FALSE;	
0525	//#TG_400.Send.STW1.%X12 := FALSE;	
0526		
0527	// Interrupt a move	
0528	IF #iMT_Pause THEN	
0529	#TG_400.Send.STW1.%X5 := FALSE; // intermediate stop, motion is	
paused and will continue when the pause is lifted		
0530	ELSE	
0531	#TG_400.Send.STW1.%X5 := TRUE; // No intermediate stop	
0532	END_IF;	
0533		
0534	// Reset the rising edge of the start bit to be read for the next task	
0535	IF #TG_400.Receive.ZSW1.%X12 OR ABS(#wStartMove - #TG_400.Receive.ZSW2) >	
62 THEN //if start command has been acknowledged, or >62 telegram transmissions		
0536	#bStartMoveDone := TRUE; // the process to start the move has been	
completed		

Totally Integrated Automation Portal		
0537	#TG_400.Send.STW1.%X6 := FALSE; //turn off the start bit	
0538	END_IF;	
0539	// Move	
0540	IF #iStart_Move AND NOT #bStartMove THEN	
0541	#TG_400.Send.MDI_MOD := #iMT_MoveType; // 1 = absolute, 0 = relative	
0542	#TG_400.Send.SATZANW := #iMTNum; // 0x8000 = direct motion task,	
	otherwise select a motion task from the AKD	
0543	#TG_400.Send.MDI_TARPOS := #iCmdPos.%W1;	
0544	#TG_400.Send.MDI_TARPOS_1 := #iCmdPos.%W0;	
0545	#TG_400.Send.MDI_VELOCITY := #iCmdVel.%W1;	
0546	#TG_400.Send.MDI_VELOCITY_1 := #iCmdVel.%W0;	
0547	#TG_400.Send.MDI_ACC := #iCmdAcc;	
0548	#TG_400.Send.MDI_DEC := #iCmdDec;	
0549	#TG_400.Send.STW1.%X4 := TRUE; // Do not reject traversing task	
0550	#TG_400.Send.STW1.%X5 := TRUE; // No intermediate stop	
0551	#TG_400.Send.STW1.%X6 := TRUE; // Start the move	
0552		
0553	#bStartMove := TRUE; // the process to start the move has been started	
	but not completed	
0554	#bStartMoveDone := FALSE; // the process to start the move has not	
	been completed	
0555	#wStartMove := #TG_400.Receive.ZSW2;	
0556	END_IF; //If Start move	
0557		
0558	END_IF; //If Start Jog	
0559	END_IF; //If Start Homing	
0560	END_IF; // If Stop	
0561	END_IF; // If ZSW1 bit2 Operation Enabled	
0562	END_IF; // If ZSW1 bit1 Switched On	
0563	END_IF; // If ZSW1 bit0 Ready for Switch On	
0564	END_IF; // If ZSW1 bit6 Switch-On Inhibited	
0565		
0566	// If the axis gets disabled all the corresponding motion signals have to be reset	
0567	ELSE	
0568	#bSWEnabled := FALSE;	
0569		
0570	#TG_400.Send.STW1.%X0 := FALSE;	
0571	#TG_400.Send.STW1.%X1 := FALSE;	
0572	#TG_400.Send.STW1.%X2 := FALSE;	
0573	#TG_400.Send.STW1.%X3 := FALSE;	
0574	#TG_400.Send.STW1.%X4 := FALSE;	
0575	#TG_400.Send.STW1.%X5 := FALSE;	
0576	#TG_400.Send.STW1.%X6 := FALSE;	
0577	#TG_400.Send.STW1.%X8 := FALSE;	
0578	#TG_400.Send.STW1.%X9 := FALSE;	
0579	#TG_400.Send.STW1.%X11 := FALSE;	
0580	#TG_400.Send.STW1.%X12 := FALSE;	
0581	END_IF; //If Enable	
0582	END_IF; // ZSW1 bit9 PLC has control	
0583	ELSE	
0584	#bPNWriteError := TRUE;	
0585	END_IF; //If #wPNWriteStatus = 0	
0586		
0587	// Read drive faults and warning on occurens	
0588	IF #oFault THEN	
0589	// If alarm is not read yet: start read	
0590	IF NOT #wSDOStatus.%X6 AND NOT #bSDOStatusReadActive THEN	
0591	#wSDOStatus := 64;	
0592	#wSDOAddress := 16#09AD; // Read PNU 2477 (DRV.FAULT1)	
0593	#bSDOStatusReadActive := TRUE;	
0594	END_IF;	
0595		
0596	// If warning is not read yet: start read	
0597	ELSIF #oWarning THEN	
0598	IF NOT #wSDOStatus.%X5 AND NOT #bSDOStatusReadActive THEN	
0599	#wSDOStatus := 32;	
0600	#wSDOAddress := 16#0AE7; // Read PNU2791 (DRV.WARNING1)	
0601	#bSDOStatusReadActive := TRUE;	
0602	END_IF;	
0603	ELSE	
0604	IF NOT #bSDOStatusReadActive THEN	
0605	#wSDOStatus := 20;	
0606	END_IF;	
0607	END_IF; //IF #oFault	
0608		
0609	// start the reading signal	
0610	IF #wSDOStatus >= 32 THEN	
0611	// Always clean out old values first	
0612	IF #bSDOWriteBusy THEN	
0613	#bSDOReq := FALSE;	
0614	ELSE	
0615	FILL_BLK(IN := 0,	
0616	COUNT := 64,	
0617	OUT => #bySDODataArray[0]);	
0618	#bSDOReq := TRUE;	
0619	#bSDOWriteDone := FALSE;	
0620	END_IF;	
0621	// Start the read cylce to the corrspoding PNU	

Totally Integrated Automation Portal

```
0622     IF #wSDOStatus = 32 OR #wSDOStatus = 64 THEN
0623         #bySDODataArray[0] := #wSDOStatus.%B0;
0624         #bySDODataArray[1] := 16#01;
0625         #bySDODataArray[2] := 16#00;
0626         #bySDODataArray[3] := 16#01;
0627
0628         #bySDODataArray[4] := 16#10;
0629         #bySDODataArray[5] := 16#01;
0630         #bySDODataArray[6] := #wSDOAddress.%B1;
0631         #bySDODataArray[7] := #wSDOAddress.%B0;
0632         #bySDODataArray[8] := 16#00;
0633         #bySDODataArray[9] := 16#00;
0634
0635         #SDOWrite(REQ := #bSDOReq,
0636                 ID := #iID,
0637                 INDEX := 47,
0638                 LEN := 14,
0639                 BUSY => #bSDOWriteBusy,
0640                 DONE => #bSDOWriteDone,
0641                 ERROR => #bSDOWriteError,
0642                 RECORD := #bySDODataArray);
0643     IF #bSDOWriteDone THEN
0644         #wSDOStatus := #wSDOStatus + 1;
0645     END_IF;
0646 END_IF; //IF #wSDOStatus = 32 OR #wSDOStatus = 64
0647
0648 // read the value back
0649 IF #wSDOStatus = 33 OR #wSDOStatus = 65 THEN
0650     #SDORead(REQ := #bSDOReq,
0651             ID := #iID,
0652             INDEX := 47,
0653             MLEN := 14,
0654             BUSY => #bSDOReadBusy,
0655             VALID => #bSDOReadDone,
0656             ERROR => #bSDOReadError,
0657             RECORD := #bySDODataArray);
0658 IF #bSDOReadDone THEN
0659     IF #bySDODataArray[4] = 66 THEN // if the read signal is a Word use Byte 6&7
0660         #wSDOReadValue.%B0 := #bySDODataArray[7];
0661         #wSDOReadValue.%B1 := #bySDODataArray[6];
0662     ELSE // if the read signal is a DWord use Byte 8&9
0663         #wSDOReadValue.%B0 := #bySDODataArray[9];
0664         #wSDOReadValue.%B1 := #bySDODataArray[8];
0665     END_IF;
0666     IF #wSDOReadValue = 0 THEN // If the read value is zero repeat the read cycle
0667         #wSDOStatus := #wSDOStatus - 1;
0668     ELSE
0669         #wSDOStatus := #wSDOStatus + 1;
0670         #bSDOStatusReadActive := FALSE;
0671     END_IF;
0672 END_IF; //IF #bSDOReadDone
0673 END_IF; //IF #wSDOStatus = 33 OR #wSDOStatus = 65
0674 END_IF; //IF #wSDOStatus >= 32
0675 END_IF; //IF NOT #iInit AND #wSDOStatus > 19
0676
0677 // check for rising edge for the input parameters
0678 IF NOT #iReset AND #bResetDone THEN
0679     #bResetDone := FALSE;
0680 END_IF;
0681 IF NOT #iStart_Homing AND #bHomeStarted THEN
0682     #bHomeStarted := FALSE;
0683 END_IF;
0684 IF NOT #iStart_Move AND #bStartMove AND #bStartMoveDone AND NOT #TG_400.Receive.ZSW1.%X12 THEN
0685     #bStartMove := FALSE;
0686     #bStartMoveDone := FALSE;
0687 END_IF;
0688
0689 // fault detection/handling within the bloc
0690 IF #bPNReadError OR #bPNWriteError OR #bSDOReadError OR #bSDOWriteError THEN
0691     #oError := TRUE;
0692     IF #bPNReadError THEN
0693         #oStatus := #wPNReadStatus; //indicates internal FB error
0694     END_IF;
0695     IF #bPNWriteError THEN
0696         #oStatus := #wPNWriteStatus; //indicates internal FB error
0697     END_IF;
0698 ELSIF #wSDOStatus > 19 THEN
0699     #oError := FALSE;
0700 END_IF;
0701
```

Symbol	Address	Type	Comment
#bEstop		Bool	
#bHomeStarted		Bool	
#bPNReadError		Bool	
#bPNWriteError		Bool	
#bResetDone		Bool	
#bSDOReadBusy		Bool	

Totally Integrated Automation Portal			
Symbol	Address	Type	Comment
#bSDOReadDone		Bool	
#bSDOReadError		Bool	
#bSDOReq		Bool	
#bSDOStatusReadActive		Bool	
#bSDOWriteBusy		Bool	
#bSDOWriteDone		Bool	
#bSDOWriteError		Bool	
#bStartMove		Bool	
#bStartMoveDone		Bool	
#bSWEnabled		Bool	
#bySDODataArray		Array	
#iCmdAcc		Word	commanded acceleration for jog or direct motion task
#iCmdDec		Word	commanded deceleration for jog or direct motion task
#iCmdPos.%W0		Word	commanded position for motion task or homing
#iCmdPos.%W1		Word	commanded position for motion task or homing
#iCmdVel		DWord	commanded velocity for jog or direct motion task
#iCmdVel.%W0		Word	commanded velocity for jog or direct motion task
#iCmdVel.%W1		Word	commanded velocity for jog or direct motion task
#iCmdVel.%X31		Bool	commanded velocity for jog or direct motion task
#iEStop		Bool	immediate quick stop and disable
#iID		HW_IO	Hardware identifier
#iInit		Bool	initialize config for TG400
#iMT_MoveType		Word	motion task control word 0 = relative; 1 = absolute
#iMT_Pause		Bool	pause active motion task
#iMTNum		Word	motion task number 0x8000 = direct motion task
#iReset		Bool	reset drive faults
#iStart_Homing		Bool	start homing
#iStart_Jog		Bool	start jog
#iStart_Move		Bool	start motion task
#iStop		Bool	stop all movement
#iSW_Enable		Bool	enable/disable
#oCurrent		Word	actual current x * DRV.IPEAK / 2^14
#oEnabled		Bool	Drive power on
#oError		Bool	fault in function block or communication
#oFault		Bool	drive fault
#oHomed		Bool	axis homed
#oInitDone		Bool	initialization status - 0 while configuring TG400, 1= done
#oInPos		Bool	axis within in-position window
#oMoving		Bool	axis enabled and moving
#oPos.%W0		Word	actual position
#oPos.%W1		Word	actual position
#oStatus		DWord	ZSW1
#oStatus.%W0		Word	ZSW1
#oStatus.%W1		Word	ZSW1
#oVel		Word	actual velocity x * 12000 rpm / 2^15
#oWarning		Bool	drive warning
#SDORead		Multi_SFB	
#SDOWrite		Multi_SFB	
#TG_400.Receive		Struct	
#TG_400.Receive.ITIST_GLATT		Int	Actual current
#TG_400.Receive.NIST_A		Int	Actual velocity
#TG_400.Receive.XIST_A		Int	Actual position
#TG_400.Receive.XIST_A_1		Int	Actual position
#TG_400.Receive.ZSW1		Int	Status word 1
#TG_400.Receive.ZSW1.%X0		Bool	Status word 1
#TG_400.Receive.ZSW1.%X1		Bool	Status word 1
#TG_400.Receive.ZSW1.%X2		Bool	Status word 1
#TG_400.Receive.ZSW1.%X3		Bool	Status word 1
#TG_400.Receive.ZSW1.%X6		Bool	Status word 1
#TG_400.Receive.ZSW1.%X7		Bool	Status word 1
#TG_400.Receive.ZSW1.%X9		Bool	Status word 1
#TG_400.Receive.ZSW1.%X10		Bool	Status word 1
#TG_400.Receive.ZSW1.%X11		Bool	Status word 1
#TG_400.Receive.ZSW1.%X12		Bool	Status word 1
#TG_400.Receive.ZSW1.%X13		Bool	Status word 1
#TG_400.Receive.ZSW1.%X14		Bool	Status word 1
#TG_400.Receive.ZSW2		Int	Status word 2
#TG_400.Send		Struct	
#TG_400.Send.HOME_DIST		Int	Home distance
#TG_400.Send.HOME_DIST_1		Int	Home distance
#TG_400.Send.MDI_ACC		Int	MDI motion task acceleration
#TG_400.Send.MDI_DEC		Int	MDI motion task deceleration
#TG_400.Send.MDI_MOD		Int	MDI motion task control word
#TG_400.Send.MDI_TARPOS		Int	MDI motion task target position
#TG_400.Send.MDI_TARPOS_1		Int	MDI motion task target position
#TG_400.Send.MDI_VELOCITY		Int	MDI motion task velocity
#TG_400.Send.MDI_VELOCITY_1		Int	MDI motion task velocity
#TG_400.Send.SATZANW		Int	Motion task selection
#TG_400.Send.STW1		Int	Control word 1
#TG_400.Send.STW1.%X0		Bool	Control word 1
#TG_400.Send.STW1.%X1		Bool	Control word 1
#TG_400.Send.STW1.%X2		Bool	Control word 1
#TG_400.Send.STW1.%X3		Bool	Control word 1
#TG_400.Send.STW1.%X4		Bool	Control word 1
#TG_400.Send.STW1.%X5		Bool	Control word 1

Totally Integrated Automation Portal				
Symbol	Address	Type	Comment	
#TG_400.Send.STW1.%X6		Bool	Control word 1	
#TG_400.Send.STW1.%X7		Bool	Control word 1	
#TG_400.Send.STW1.%X8		Bool	Control word 1	
#TG_400.Send.STW1.%X9		Bool	Control word 1	
#TG_400.Send.STW1.%X10		Bool	Control word 1	
#TG_400.Send.STW1.%X11		Bool	Control word 1	
#TG_400.Send.STW1.%X12		Bool	Control word 1	
#TG_400.Send.STW1.%X13		Bool	Control word 1	
#TG_400.Send.STW2		Int	Control word 2	
#wHomingStartPoint		Word		
#wPNReadStatus		Word		
#wPNWriteStatus		Word		
#wSDOAddress		Word		
#wSDOAddress.%B0		Byte		
#wSDOAddress.%B1		Byte		
#wSDOReadValue		Word		
#wSDOReadValue.%B0		Byte		
#wSDOReadValue.%B1		Byte		
#wSDOStatus		Word		
#wSDOStatus.%B0		Byte		
#wSDOStatus.%X5		Bool		
#wSDOStatus.%X6		Bool		
#wStartMove		Int		
#wStartReset		Int		